Journal of Computer and Knowledge Engineering, Vol. 9, No. 1, 2026. (19-36)

Ferdowsi University of Mashhad

Journal of Computer and Knowledge Engineering

https://cke.um.ac.ir

Information and Communication Technology Association of Iran

Fraud Detection Using Temporal Sequence Embedding and Cluster-Weighted LSTM Models in Card Transactions*

Research Article

Samiyeh Khosravi[1], Mehrdad Kargari[2], Babak Teimourpour[3], Mohammad Talebi[4], Abdollah Eshghi[5]

10.22067/cke.2025.92170.1150

Abstract *Traditional fraud detection models often overlook the sequential and temporal relationships between transactions, which can be crucial for identifying fraudulent activities. To address this, a new data-to-graph mapping approach is proposed, transforming user data into a transaction graph and constructing a bipartite graph with source and target nodes. The main goal is to leverage the temporal order of transactions to capture changes effectively and identify distinct fraud patterns. The method begins by creating a weighted graph based on transaction amounts and their temporal sequence. For feature extraction, the Probabilistic FraudWalk method—an advanced version of the traditional FraudWalk algorithm—is used. This method enhances the random walk process by incorporating probability-based neighbor selection, dynamically choosing the next node based on the probability distribution of common neighbors. To balance the dataset, the Synthetic Minority Over-sampling Technique (SMOTE) is combined with the Edited Nearest Neighbors (ENN) method, forming SMOTE-ENN. To reduce information conflict, data is clustered using K-means clustering. A weighted Long Short-Term Memory (LSTM) model is then trained on each cluster, with weights determined by the minimum distance between samples of different classes within the same cluster. The proposed LSTM model demonstrates superior performance on benchmark datasets, effectively detecting fraud in real-world card-to-card transactions. This approach enhances the security of financial information for banks and financial institutions, showing that incorporating temporal and sequential data significantly improves fraud detection accuracy and reliability.*

Key Words Fraud detection, Bipartite graph, Node embedding, Weighted LSTM, Source neighbor sequence, Target neighbor sequence, Temporal node embedding

1. Introduction

In recent years, due to the growth of e-commerce and the use of online payments, the financial industry has witnessed a significant increase in the volume and complexity of financial transactions, leading to a surge in fraudulent activities [1],[2]. In the 2022 report from the Canadian Anti-Fraud Centre (CAFC), it was noted that over 91,190 fraud incidents were documented, with 57,055 individuals suffering losses that surpassed $531 million [3]. This highlights the fact that financial fraud can result in the loss of large amounts of money and undermine trust in financial institutions and systems. This recent rise in fraudulent activities underscores the crucial need for effective fraud detection. And since financial domains include such as cryptocurrency, online payment transactions, taxation, medical insurance, and credit cards. There are variants of fraud, among which this study is focused on card transaction. The increasing credit card usage has brought about a constant increase in fraudulent transactions [3].

Fraudulent credit card transactions have severely impacted the financial industry. According to a recent study, credit card fraud resulted in losses of approximately 27.85 billion dollars in 2018, marking a 16.2% rise from the 23.97 billion dollars lost in 2017. Projections suggest that these losses could escalate to 35 billion dollars by 2023 [4]. Hence, this recent rise in fraudulent activities underscores the crucial need for effective fraud detection.

* Manuscript received February 13, 2025, Revised, July 27, 2025, Accepted August 18, 2025.

[1] Assistant professor, Department of Computer Engineering, Faculty of Electrical and Computer Engineering, University of Birjand, Birjand, Iran. **Email**: skhosravi@birjand.ac.ir

[2] Corresponding author, Associate professor, Faculty of Industrial and systems engineering,Tarbiat Modares University, Tehran, Iran. **Email:** m_kargari@modares.ac.ir

[3] Associate professor, Faculty of Industrial and systems engineering,Tarbiat Modares University, Tehran, Iran. **Email:** b.teimourpour@modares.ac.ir

[4] Associate professor , Faculty of Management, Imam Sadiq University, Tehran, Iran. **Email:** Mohammad63.mt@gmail.com

[5] PhD in Information Technology, Faculty of Industrial and systems engineering,Tarbiat Modares University, Tehran, Iran.

Therefore, identifying this form of fraud early can avert its substantial financial impact.

Traditional approaches for fraud detection are included rule-based systems and classical machine learning techniques [5].In general, rule - based systems rely on human-designed rules with expert knowledge to assess the likelihood that fraud has occurred, which cannot perform well in complex environments. Also, the fixed rules limit the algorithm's ability to adapt to dynamic fraud patterns [6].To address the limitations of rule-based systems, new transaction monitoring methods utilizing data science and machine learning techniques have been introduced [7],[8].

The review of existing research suggests that a majority of scholars in this area tend to employ a range of data mining techniques [7],[9]. Classical machine learning techniques achieving statistical features from transaction attributes such as time, location, and amount is feasible. However, incorporating unstructured data is challenging to extract. Furthermore, struggle to handle high-dimensional data, non-linear relationships, complex patterns that are common in financial fraud detection, temporal dependence between transactions, capturing the interaction between transactions presents difficulties [1],[2]. The origins of these issues can be traced to the swift advancement of contemporary technologies, which has rendered traditional methods ineffective against new fraudulent techniques. Specifically, these conventional approaches rely on descriptive statistics [1],[2] as features to incorporate historical data [3]. These statistics typically describe credit card transactions using factors such as transaction time, amount, and merchant category, but often fail to account for precise sequential information.

Put differently, interactions between users can occur on multiple levels simultaneously. User can interact with each other and deposit or withdraw their money through several intermediary nodes cards and also communications between users can change over time. This means that, the intermediary nodes between user and the way they interact can change over time.

So, the time sequence in user's communications with each other can be a key factor, and different patterns can appear by changing the time sequence. For example, the descriptive statistics is the number of transactions or the total amount spent by the cardholder in the last 24 hours for a merchant category or country. In these descriptive features, communications between senders and receivers are extracted without considering the time sequence of these communications. For instance, person A deposits amount to person B's account, and then person B deposits amount b to person C's account. In this case, the connection between A and C is through B, and this should be considered in extracting transaction fraud, and not only the source, i.e., depositor and the target, i.e., receiver should be considered.

Therefore, these embedded connections between users play an important role in making fraud undetectable.

In this study, the occurrence of fraud in card transaction of Iranian banks, which are the most vital components of the country's economy, has been studied. Consequently, a novel approach model is proposed, which consists of two phases. In first phase, the features of communications between users are extracted and are used in the fraud detection process. In this way, to extract the embedded features between the source and target of transactions, the mapping of user's information into a graph is utilized, and a graph of transactions is created between the source user as the source Node and the destination user as the target Node. This graph appears as a bipartite graph, and the edges are marked with timestamps and amounts. The two concepts of source neighbor sequence and target neighbor sequence are exploited for embedding communication between users.

Then, by forming a weighted graph based on these two types of sequences, we perform the probabilistic fraud walk and the embedding features are extracted for each node. The second phase solves the conflict is created in extracting embedded information with each other, which makes them either ignored or ineffective in the learning process. For this purpose, a multi-model approach is used. Hence, training data is first clustered so that each cluster maintains its information. Then, , due to ability to learn intricate features and model nonlinear relationships in the deep learning models, making them well - suited for identifying subtle, hidden patterns indicative of fraudulent activities, is utilized to train each cluster. Each cluster discovers its information separately.

In experiments, we compared the proposed method on benchmark datasets such as European and Brazilian in terms of F1-score, Macro-F1, Micro-F1 and AUC, which shows the superiority of the proposed method over other existing methods. Moreover, according to the research case study dataset of card-to-card banking transactions of Iranian customers, the results demonstrate the obvious superiority of the proposed method.

The contributions of this paper are as follows:

Temporal Sequence Embedding: Unlike traditional methods that rely on descriptive statistics of transactions, this approach explicitly models the temporal sequence of transactions. By transforming user data into a transaction graph, constructing a bipartite graph with source and target nodes, and utilizing source and target neighbor sequences, the method captures the dynamic interactions between users over time. This is a key improvement, as the time sequence in user communications can reveal distinct fraud patterns.

Cluster-Weighted LSTM: To address the issue of information conflict, where different data points or areas with varying densities can interfere with the learning process, the training data is first clustered using K-means. This allows for the learning of local information of data with similar behavior.

A weighted LSTM model is then trained on each cluster, with weights determined by the minimum distance between samples of different classes within the same cluster. This approach allows each cluster to learn its information separately, effectively capturing the subtle and hidden patterns indicative of fraudulent activities. The weighting mechanism enhances the LSTM's ability to classify marginal samples, which are often the most challenging to identify, thus leading to improved accuracy in fraud detection.

Enhancement of the Random Walk: The proposed method enhances the random walk by incorporating probability-based neighbor selection and temporal order consideration. This is an improvement because the

traditional FraudWalk method selects the next node randomly during walks and does not consider intermediary nodes. By prioritizing the selection of the next node based on the highest number of common neighbors within a temporal window, the Probabilistic FraudWalk generates more effective walks and embeddings, and is more sensitive to complex patterns of collaboration. By considering temporal constraints and the probability distribution of common neighbors, the method ensures a more comprehensive representation of the network.

The reminder of the paper is organized as follows. Section 2 introduces the related literature. Section 3 explains the concepts related to the extraction of embedded temporal communications. Section 4 introduces the proposed method, and Section 5 describes the real-world and benchmark datasets and presents the experimental results and discussion. Finally, conclusions are given in Section 6.

2. RELATED WORK

The increasing prevalence of credit card fraud, driven by the rise in online transactions, has spurred extensive research into advanced fraud detection techniques. This section systematically categorizes prior studies into three main approaches: traditional machine learning, deep learning, and graph-based methods. Each category is discussed, followed by a comparative table summarizing the key features of the reviewed studies.

2.1 Traditional Machine Learning Approaches

Traditional machine learning methods, such as logistic regression and random forests, have been widely applied to fraud detection due to their interpretability and effectiveness with structured data. Wu et al. [10] utilized an advanced management information system combining logistic regression, random forests, and Long Short-Term Memory (LSTM) networks to detect Mastercard fraud during the COVID-19 pandemic. Their approach leveraged statistical features but struggled with capturing temporal dependencies. Similarly, Esenogho et al. [4] proposed a neural network ensemble with feature engineering, achieving improved performance on credit card fraud datasets by incorporating domain-specific features. However, these methods often fail to model complex relationships and temporal dependencies.

2.2 Deep Learning Approaches

Deep learning methods have gained popularity in fraud detection due to their ability to learn complex and non-linear patterns in transactional data. Xie et al. [11] developed a time-aware historical-attention-based LSTM (TH-LSTM) model that automatically detected fraudulent patterns by capturing behavioral changes induced by sequential user transactions. Roseline et al. [12] proposed an LSTM-RNN with an attention mechanism, enhancing fraud detection performance by leveraging intricately connected feature vectors. Guo et al. [13] introduced a Historical Attention-based and Interactive LSTM (HAInt-LSTM) model to identify sequential patterns and flag deviations as potential fraud. Agarwal et al. [14] employed a hybrid CNN-BiLSTM-Attention model, combining convolutional neural networks (CNN) and bidirectional LSTM with an attention mechanism, achieving commendable accuracy in detecting fraudulent activities. Similarly, Fakiha [15] proposed an LSTM-Attention model that improved performance by selecting relevant features and transaction sequences. Zioviris et al. [16] presented a multistage deep learning framework using autoencoders and deep convolutional neural networks for fraud detection through feature selection and latent representation. Raval et al. [17] integrated explainable artificial intelligence (XAI) with LSTM, developing an explainable LSTM (X-LSTM) model that enhanced transparency and accuracy in identifying fraud patterns.

2.3 Graph-Based Approaches

Graph-based methods focus on the relationships and interactions between users in transactional networks, making them suitable for modeling complex fraud patterns. Wang et al. [18] proposed a semi-supervised attentive graph neural network (SemiGNN) with a hierarchical attention mechanism, utilizing both labeled and unlabeled data to identify factors contributing to fraud. Jiang et al. [19] extracted source and target neighbor sequences from a temporal bipartite network and integrated the Hawkes process into LSTM to enhance historical influence learning. Liu et al. [20] developed a Hierarchical Attention-based Graph Neural Network (HA-GNN) with weighted adjacency matrices to detect fraudulent activities while mitigating camouflage risks. Wang et al. [21] proposed a Community-based Framework with Attention mechanism for analyzing large-scale Heterogeneous graphs (C-FATH), improving detection accuracy by filtering structurally inconsistent nodes. Lastly, the FraudWalk method in [22] extracted connections between source and target nodes in a bipartite graph but was limited by random node selection and lack of intermediary node consideration. Our proposed method overcomes these limitations by incorporating probability-based neighbor selection and temporal sequence consideration, enhancing fraud detection performance.

Recent studies have further advanced temporal and imbalanced data handling in fraud detection. Imran and Yakoob [23] proposed a credit card fraud detection system that integrates Synthetic Minority Oversampling Technique (SMOTE) with an attention mechanism and dual LSTM layers to model long-term dependencies in transaction sequences, achieving high accuracy in predicting fraudulent transactions.. Kim et al. [24] proposed a dynamic graph convolutional network (DGCN) that incorporates temporal constraints and attention mechanisms to detect evolving fraud patterns in real-time transaction streams, offering a robust solution for dynamic financial environments.

Table 1 compares the key features of the reviewed studies, including the model used, dataset, key techniques, and evaluation metrics. This table and systematic categorization provide a comprehensive overview of existing methods, highlighting the strengths and limitations of each approach. Our proposed method, by integrating temporal sequence embedding, cluster-weighted LSTM, and probability-based neighbor selection, overcomes the limitations of prior methods and improves fraud detection performance.

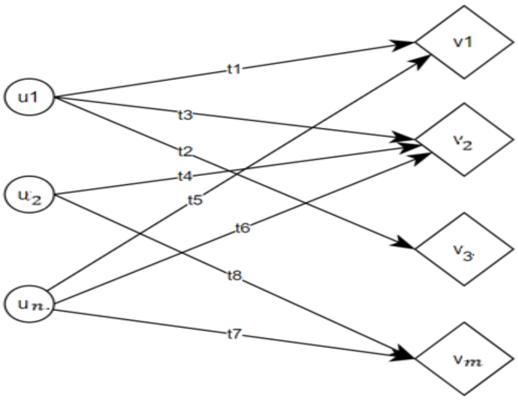


Figure 1. Temporal bipartite network.

TABLE 1
Comparison of key features of the reviewed studies

Reference	Model	Dataset	Key Techniques	Evaluation Metrics
[4]	Neural Network Ensemble	Credit Card Data	Feature Engineering, Neural Network Ensemble	Accuracy, F1-Score
[10]	Logistic Regression, Random Forest, LSTM	Mastercard Data	Information System, Deep Learning	Accuracy, AUC
[11]	TH-LSTM	Transactional Data	Time-aware Historical Attention, LSTM	F1-Score, AUC
[12]	LSTM-RNN	Credit Card Data	Recurrent Network, Attention Mechanism	Accuracy, F1-Score
[13]	HAInt-LSTM	Sequential Behavioral Data	Historical Attention, LSTM	F1-Score, Accuracy
[14]	CNN-BiLSTM-Attention	Credit Card Data	Convolutional Network, Bidirectional LSTM, Attention	Accuracy, F1-Score
[15]	LSTM-Attention	Credit Card Data	Feature Selection, Attention, LSTM	F1-Score, AUC
[16]	Autoencoder, Deep CNN	Transactional Data	Multistage Framework, Feature Selection	F1-Score, AUC
[17]	X-LSTM	Credit Card Data	Explainable AI, LSTM	Accuracy, F1-Score
[18]	SemiGNN	Transactional Data	Semi-supervised Graph Network, Hierarchical Attention	F1-Score, AUC
[19]	Hawkes-LSTM	Temporal Bipartite Network	Hawkes Process, Historical Attention, LSTM	F1-Score, AUC
[20]	HA-GNN	Transactional Data	Hierarchical Attention-based Graph Network, Weighted Adjacency	F1-Score, AUC
[21]	C-FATH	Heterogeneous Graphs	Community-based Framework, Attention Mechanism	F1-Score, AUC
[22]	FraudWalk	Bipartite Network	Source and Target Sequence Extraction	F1-Score, AUC

PRELIMINARIES

In the real world, the communications between the financial accounts of the sender and receiver at different times form a temporal bipartite network, where sender accounts act as source users, receiver accounts act as target users, and a directional edge with a timestamp indicates that a specified amount has been sent from the sender's account to the receiver's account.

Definition of temporal bipartite graph

A temporal bipartite graph is a graph that has edges between source and target nodes along with temporal connections between nodes. A temporal bipartite graph is defined as $G =< U, V, E >$, where $U = \{u_1, u_2, \dots, u_n\}$ represents the source nodes, $V = \{v_1, v_2, \dots, v_m\}$ represents the target nodes, and E is a set of edges formed through the communications between U and V at different times t [19]. , consider the temporal bipartite graph shown in Figure 1.

With the help of this graph, the dynamic changes in the graph can be clearly displayed. The target nodes can be expressed as a sequence of nodes arranged in ascending order, called the target neighbor sequence, as follows:

Definition of target neighbor sequence

According to a source node u in a temporal bipartite graph and its target nodes $TN(u)$, the target neighbor sequence for the user u is displayed as $TNS(u) = [(t_1, v_1), (t_2, v_2), \dots, (t_n, v_n)]$ in time order. Each tuple shows the communication between the source node u and the target neighbor node $v_i \in TN(u)$ at time t_i.

In other words, connections between source nodes can be achieved by any target node, so that for each target node, a time sequence of source nodes that have interacted with the target node at different times is obtained. This sequence of source nodes represents the embedded communications between these nodes [19].

Definition of source neighbor sequence

According to a target node v in a temporal bipartite graph and its related source nodes $SN(v)$, the source neighbor sequence for the user v is displayed as $SNS = [(t_1, u_1), (t_2, u_2), \dots, (t_m, u_m)]$ in time order, in which each tuple shows the connection between the target node v and the source neighbor node $u_i \in SN(v)$ at time t_i.

3. PROPOSED METHOD

In the proposed method, we need to obtain different features of users and their behavior in interaction with each other to distinguish fraudulent user from normal user. Many features are embedded and can be extracted by analyzing users' behavior over time via deep learning methods.

For example, consider the network shown in Figure 2. In this network, a_i, representing the financial transaction amount, and t_i, representing the transaction time, have great impacts on the communication of nodes.

The general algorithm of the proposed method is as follows.

The first step, Construct source neighbor sequences and the second step Construct target neighbor sequences.

Then, we created a weighted directed graph. In the fourth step, extract node embedding for each node of the graph based on the Probabilistic Fraud Walk. Next, split the dataset into training and testing samples. The sixth step, Oversample training samples based on SMOTE-ENN technique. Afterwards Cluster training samples and the next step train a weighted LSTM model for each cluster based on the distance to the opposite sample in a different class. Finally predict the label of each testing sample based on the corresponding LSTM trained for its cluster. The proposed method is shown in Figure 3.

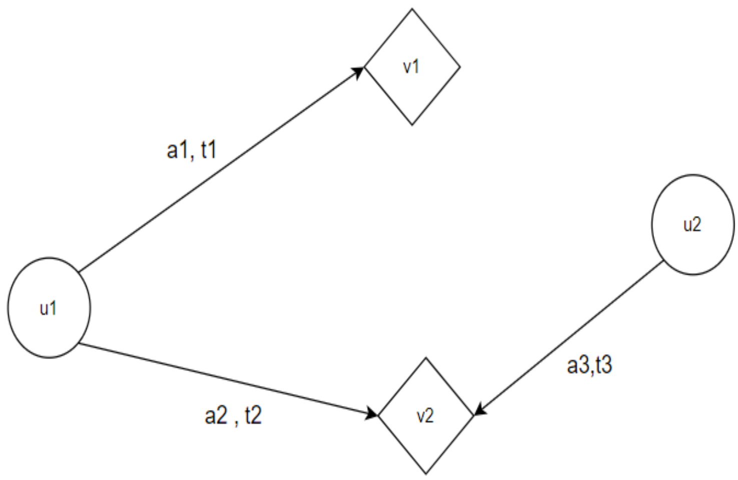


Figure 2. Card-to-card transfer network.

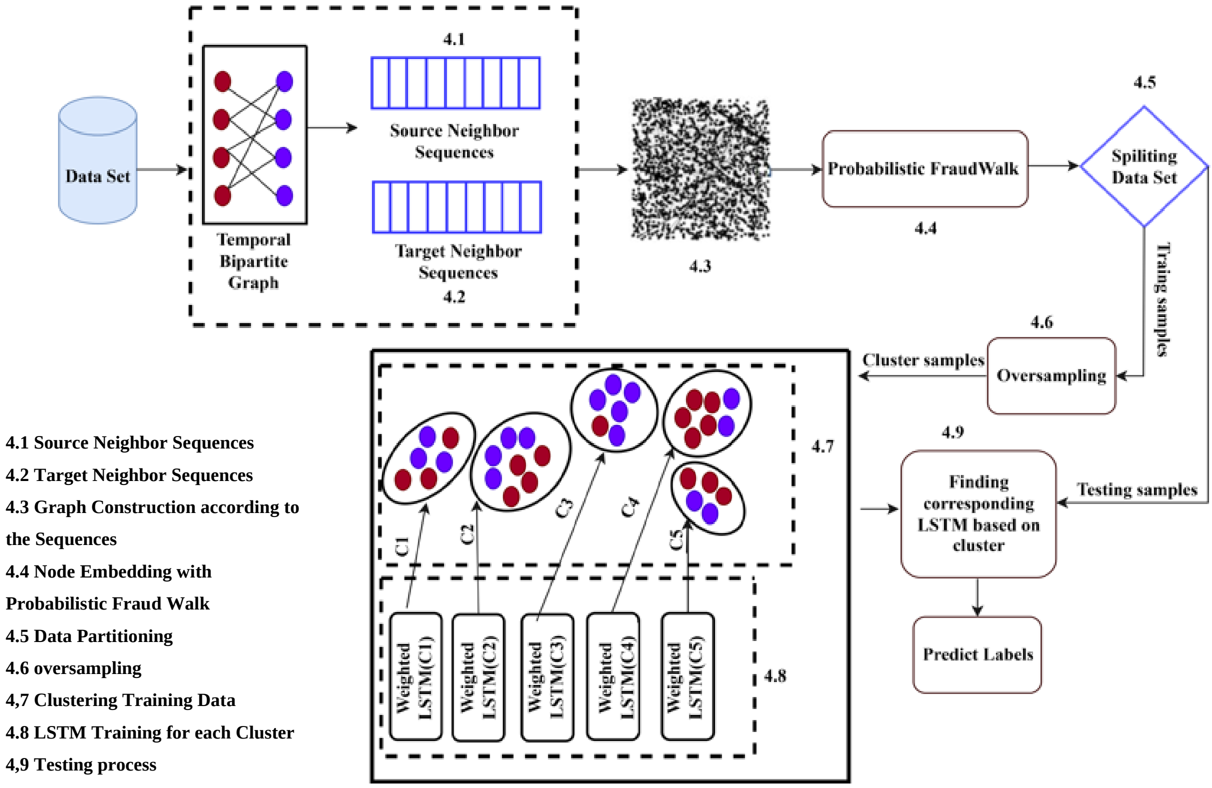


Figure 3. Proposed method: temporal sequence embedding and Cluster-Weighted LSTM Models in Card Transactions

The steps of the proposed method are described below.

4.1. Constructing a source neighbor sequence

To construct this sequence, first, for each target node, the corresponding source nodes are listed in chronological order. Then, the sequences obtained for each target node are listed in the source neighbor sequence set.

This sequence of source nodes shows the communications of these nodes regarding their interactions with the target nodes. For example, suppose we have the following sequences:

$$\{u_1, u_2, u_{10}, u_{20}\}$$
$$\{u_1, u_2\}$$
$$\{u_1, u_2, u_3, u_5\}$$

These sequences show that usually the two nodes u_1 and u_2 appear next to each other, which means that they have been connected with a series of common target nodes in a consecutive time interval. Therefore, there can be an embedded connection between them.

4.2. Constructing a target neighbor sequence

In these sequences, the communications between the target nodes are specified. For example, as can be seen, two nodes v_2 and v_{10} are placed next to each other most of the time, and this

indicates that many source nodes have interacted with v_{10} without interruption after interacting with v_2.

$$\{v_1, v_2, v_{10}, v_{11}\}$$
$$\{v_9, v_2, v_{10}, v_{20}\}$$

Therefore, there can be embedded communications between the two nodes v_2 and v_{10}, and we can extract this information with the help of target neighbor sequences.

4.3. Constructing a graph between nodes according to their communication

After extracting the sequences between the source nodes (source neighbor sequence) and between the target nodes (target neighbor sequence), a graph of nodes is constructed using these sequences.

According to the relation below, edges are created between the nodes in the sequence.

$$u_1, u_2, u_3, u_4, u_5$$
$$e_1 \quad e_2 \quad e_3 \quad e_4$$

$$E = \{(u_i, u_{i+1}) | u_i \in S_{node}\}$$

The weight of the edges is as follows:

$$W_{u_i,u_{i+1}} = \sum_{u \in V} (A_{u_i,v} + A_{u_{i+1},v})$$

where v is the set of common nodes between u_i and u_{i+1}, and $A_{u_i,v}$ represents the amount between the nodes u_i and v.

In this way, for any two consecutive nodes in different sequences, the desired weight is obtained from the sum of the amounts of the two nodes, and as a result, the corresponding weighted graph between the nodes is

constructed.

4.4. Generating node embeddings for graph nodes

According to the feature of time sequence between the nodes as well as the feature of amount for the nodes in the graph, the neighborhood information between the nodes is considered in this step. This neighborhood information is determined according to the paths between the two nodes. In other words, the shorter the distance or path between the two nodes, the stronger their connections with each other, and the farther this distance is, the weaker their connections will be.

Algorithm 1: Probabilistic FraudWalk

Data: Bipartite network $G = (U, V; E)$, time constraint δt, embedding size d, walk length l, window size w
Result: Matrix of node embeddings $\mathbf{X} \in R^{|G| \times d}$
// Initialize the embedding matrix
$\mathbf{X}$ = Sample From Distribution($R^{|G| \times d}$);
for $j = 0$ **to** *MaxIterations* **do**
  // Shuffle the nodes in the network
  M_j = Shuffle($U \cup V$);
  for *each* $v_i \in M_j$ **do**
    // Conduct Constrained Random Walk with probability
    $W_v i$ = Constrained Random Walk With Probability(v_i, δt, l, w);
    // Apply SkipGram
    SkipGram($\mathbf{X}$, $W_v i$);
  end
end

For constructing embedding features of nodes, we use a Probabilistic FraudWalk that reflects the addition of the probability-based neighbor selection in the Constrained Random Walk, distinguishing it from the original FraudWalk algorithm [25].

Probabilistic FraudWalk is an enhanced version of the traditional FraudWalk algorithm designed to capture nuanced interactions within a bipartite network, particularly in fraud detection scenarios. It has been presented in Algorithm 1.

In this modified approach, a novel step has been introduced during the random walk process, known as Constrained Random Walk with Probability-based Neighbor Selection. Unlike the original random walk, Probabilistic FraudWalk dynamically selects the next node to traverse based on the probability distribution of common neighbors. This strategic selection is rooted in the notion that nodes sharing more common neighbors in their respective target sequences are more likely to be relevant for capturing higher-order interactions. By integrating this probability-based mechanism, Probabilistic FraudWalk aims to enhance the algorithm's sensitivity to complex patterns of collaboration among source nodes, thereby offering a more comprehensive representation for subsequent analysis.

Algorithm 2: Constrained Random Walk with Probability-based Neighbor Selection

Data: Starting source node v_i, Temporal constraint parameter δt, Maximum walk length l
Result: Walk sequence *walk sequence*
Initialize empty sequence *walk sequence*;
Set current node *current node* to starting node v_i; Initialize walk length *walk length* to 0;
while *walk length* $< l$ **do**
  // Get the target sequence neighbors of the source node
  $T_{vi} \leftarrow$ Target Sequence Neighbors(v_i);
  // Compute probability distribution based on common nodes in target sequence neighbors
  for *each neighbor* v_k *in* T_{vi} **do**
    // Get the target sequence neighbors of the current neighbor v_k
    $T_{vk} \leftarrow$ Target Sequence Neighbors(v_k);
    Allneighbors=$\{T_{vk}\}$
    for *each neighbor* v_m *in* T_{vk} **do**
      Tvm ← Target Sequence Neighbors(vm);
      Allneighbors= Allneighbors ∪T_vm
    // Compute the number of common nodes
      C(vk) ← Count Common Nodes(Tvi, Allneighbors);
    end
  // Normalize the probability distribution

$$P(v_k) = \frac{C(vk)}{\sum^{neighbors}_{vk} C(vk)};$$

  // Sort nodes based on the normalized probability in descending order
  Sort nodes in descending order of P (vk);
  for each neighbor v_k according to the probability distribution **do**
    Check temporal constraint: Ensure $t_u i, v_k$ within $[t_{u,v} - \delta t, t_{u,v} + \delta t]$;
    if *Temporal constraint satisfied* **then**
      Add v_k to *walk sequence*;
      Update *current node* to v_k;
      Break from the loop;
    end
  end
  if No valid neighbor found then
    End the walk;
  end
  Increment walk length by 1;
End

Constrained Random Walk with Probability-based Neighbor Selection is the core innovation embedded within Probabilistic FraudWalk. This algorithm dictates the node selection process during the random walk by considering the temporal constraints on interaction events and introducing a probability distribution based on common neighbors. For each potential neighbor, the algorithm evaluates the temporal constraints to ensure that interactions occurred within a specified time window. The probability distribution is computed by measuring the commonality of target sequence neighbors between the current node and its potential neighbors. After normalization, nodes are selected in descending order of probability, emphasizing those with the most shared neighbors. This tailored approach empowers the random walk to explore paths influenced by the network's underlying structure and the temporal nature of interactions, promoting a more effective representation of intricate relationships in the bipartite network.

In algorithm 2, Constrained Random Walk with Probability-based Neighbor Selection, is presented with details.

4.5 Data partitioning

In this step, 80% of benchmark datasets are considered training sets, and 20% are test sets. For the real-world dataset, 30%, 60%, and 90% of the dataset are considered training sets, and the rest are test sets. In this research, the 5-fold cross-validation output is used on the real-world dataset.

4.6. Oversampling of training data

In this step, we are using the SMOTE-ENN (Synthetic Minority Over-sampling Technique Edited Nearest Neighbors) method on the training data. The SMOTE-ENN technique is a data preprocessing algorithm designed to address class imbalance in datasets, particularly prevalent in scenarios where one class is significantly underrepresented compared to the other.

The algorithm operates in two key steps: oversampling and undersampling. In the oversampling phase, SMOTE randomly selects samples from the minority class and generates synthetic samples by connecting them to their nearest neighbors. This process effectively augments the minority class, enhancing its representation in the dataset. The generated synthetic samples are assigned the minority class label. Following the oversampling, the algorithm moves to the undersampling phase. It evaluates each synthetic and original minority class sample, assessing their nearest neighbors. If a minority class sample has more neighbors from the majority class, it is discarded. This ensures a balanced representation by eliminating redundant or potentially misclassified minority class instances. The final output is a more balanced dataset, mitigating the challenges posed by class imbalance and improving the performance of machine learning models trained on such data. It has been presented in Algorithm 3.

Algorithm 3*:* SMOTE-ENN Technique

Data: Input data
Result: Balanced dataset
Oversampling:
for i in minority class **do**
    Choose a random sample x_i from the minority class;
    Search for the K nearest neighbors of x_i;
    Generate a synthetic sample p by randomly selecting one of the K nearest neighbors q, and connect p and q to create a line segment in the feature space;
    Give the minority class label to the newly created synthetic sample; Generate successive synthetic samples as a convex combination of the two selected samples;
end
Under sampling:
for x_i S where S denotes the total number of samples x_i from the minority class **do**
    Search for the K nearest neighbors of x_i;
        if x_i has more neighbors from the other class **then**
            Discard x_i;
        end
end

4.7. Clustering of training data

After balancing the data from two classes for the training dataset, this dataset is divided into clusters in this step. This is because applying the deep learning algorithm to the data in general makes the points that are far from each other or areas with different densities work together in the LSTM learning process. This makes most of the information disappear or become less important. Therefore, it is necessary to consider local information.

Local information means information about the behavior of data together, which is different from the behavior of all data together. This local or partial information can play a significant role in the LSTM training process. Therefore, to achieve this goal, we cluster the training dataset and train a separate LSTM for each cluster. In the LSTM training process, different weights are assigned to samples in each cluster. These weights are based on the inverse of the distance to the nearest sample of the opposite class in the same cluster. The reason for this weighting is the bias of the LSTM training process of each cluster to the marginal samples within the cluster. By increasing the importance of marginal samples, the LSTM algorithm can better classify the samples in each cluster.

4.8. LSTM training for each cluster

The training algorithm for each cluster is as follows:

Algorithm 4: LSTM training for each cluster

1. Separate positive samples C_P and negative samples C_N in one cluster.

2. Compute the distances from each negative sample to its nearest positive neighbor.

3. Compute the distances from each positive sample to its nearest negative neighbor.

4. Compute the weights for each negative sample w_i based on the distance to its nearest positive neighbor j:

$$\left\{ w_i = \frac{1}{\mathrm{D_j}} \middle| \forall\ i \in C_N, \exists\ j \in C_P, j\ is\ nearest\ neighbor\ of\ i \right\}$$

5. Compute the weights for each positive sample w_j based on the distance to its nearest negative neighbor i:

$$\left\{ w_j = \frac{1}{\mathrm{D_i}} \middle| \forall\ j \in C_P, \exists\ i \in C_N, i\ is\ nearest\ neighbor\ of\ j \right\}$$

6. Combine the sample weights for positive and negative samples.

7. Train the LSTM model with weighted samples.

4.9. Testing process

In this step, the cluster label is obtained for each input test sample. Then, the trained LSTM model associated with the test sample cluster is used to classify the test sample, and as a result, the test sample label is determined. In algorithm 5 is presented.

Algorithm 5: Testing process

Determine the cluster number of a sample test.

Predicted_class = LSTM[cluster_number].predict(test)

In the next step, the results of the experiments are shown.

5. EXPERIMENTS

In this section, we conduct fraud detection experiments on real-world and benchmark datasets. In this paper, codes are simulated via Python software. The system used for experimental evaluation is a GeforceRTX 2080 Ti with a CPU 8 cores and 24 GB of RAM and 150 GB of HDD.

5.1. Real-world dataset

The real-world dataset comprises 1,048,575 card-to-card transactions collected from an Iranian bank over a 15-day period in late July 2022. This dataset captures a diverse set of credit card activities, including both normal and potentially fraudulent transactions, making it a valuable resource for developing and evaluating fraud detection models. Each transaction is characterized by attributes such as the source account (Primary_ID), destination account (Second_ID), transaction type (TRNS_Type), date, time, amount, terminal type, and terminal ID. Additionally, a separate label dataset associates Second_IDs with binary labels (1 for fraudulent, 0 for normal), enabling supervised learning for fraud detection. The dataset's complexity and significant class imbalance pose realistic challenges for modeling fraudulent behaviors in financial systems.

I. Transaction Dataset Overview

The transaction dataset includes 1,048,575 records, with 338,987 unique Primary_IDs (source accounts) and 16,965 unique Second_IDs (destination accounts). The transactions are associated with 1,893 normal and 1,107 fraudulent accounts (based on Second_ID labels), highlighting a significant class imbalance (approximately 0.11% fraudulent transactions). This imbalance is a common challenge in fraud detection, necessitating advanced preprocessing techniques to ensure effective model training. Table 2 illustrates the diversity of transaction attributes, including varying dates, amounts, and terminal types, which are critical for capturing temporal and behavioral patterns in fraud detection.

TABLE 2
Example of Real-World Transaction Dataset

Primary_ID	**TRNS_Type**	**Date**	**Time**	**Amount**	**Terminal_Type**	**Terminal_ID**	**Second_ID**
992e7ca6cc0d738d85b2	10	7/28/2022 0:00	1.42E+08	872000	7	13ae6270758cdba26ec1	8457468be95f8a197756
737b3293b0d5ade9ad90	10	5/6/2022 0:00	1.36E+08	1700000	1	f52c0e50a8eb31e3965c	9d53d2f9c5fe1fc7f260
737b3293b0d5ade9ad90	10	7/8/2022 0:00	1.75E+08	3400000	1	f52c0e50a8eb31e3965c	9d53d2f9c5fe1fc7f260

Column Descriptions:

- **Primary_ID**: Unique identifier for the source account (sender card).
- **TRNS_Type**: Type of transaction (e.g., card to card, Payment, or purchase).
- **Date**: Date of the transaction in YYYY-MM-DD format.
- **Time**: Time of the transaction in HH:MM:SS format.
- **Amount**: Transaction amount in Rials (presented in standard numerical format for clarity).
- **Terminal_Type**: Type of terminal used for the transaction (e.g., POS, Online, ATM).
- **Terminal_ID**: Unique identifier for the terminal where the transaction occurred.
- **Second_ID**: Unique identifier for the destination account (receiver card).

II. Label Dataset Overview

The label dataset provides binary labels for the 16,965 unique Second_IDs, with 1,107 accounts marked as fraudulent (Label = 1) and 15,858 as normal (Label = 0). This dataset enables the classification of destination accounts based on their transaction history, supporting the identification of fraudulent patterns. Table 3 showes Example of Real-World Label Dataset.

TABLE 3
Example of Real-World Label Dataset

Second_ID	Label
2bd4b22499440c9fbade	1
1b777cbc4dbbd24a82ce	1
cc1e4d4a68aefa0e0042	0
7717d2414b446e7b64c7	0
721b98a42582971495d2	0

Column Descriptions:

- **Second_ID**: Unique identifier for the destination account (receiver card).
- **Label**: Binary indicator of account status (1 for fraudulent, 0 for normal).

Table 4 provides meaningful insights into the labeling process, highlighting the distinction between fraudulent and normal accounts based on their transaction behaviors

Additional Notes:

- **Feature Details**: The dataset includes 8 attributes that capture the source and destination of transactions, transaction type, temporal information, amount, and terminal details. These features enable the modeling of complex temporal and behavioral patterns.
- **Class Imbalance**: With only 1,893 fraudulent transactions out of 1,048,575, the dataset exhibits a severe class imbalance (0.11% fraudulent), necessitating techniques like SMOTE-ENN to balance the data for effective model training.

Temporal Context: The 15-day period provides a rich temporal context for analyzing sequential transaction patterns, which is critical for detecting fraud in dynamic financial systems.

III. Data Preprocessing

To ensure data quality and compatibility with machine learning models, the dataset underwent the following preprocessing steps:

- **Data Cleaning**: Missing or invalid entries were addressed by imputing median values for numerical fields (e.g., Amount) and mode values for categorical fields (e.g., TRNS_Type, Terminal_Type). Duplicate transactions were removed.
- **Feature Encoding**: Categorical features (TRNS_Type, Terminal_Type) were converted to numerical format using one-hot encoding. Date and Time were combined into a unified timestamp feature to facilitate temporal analysis.
- **Normalization**: Transaction amounts, ranging from 136,000,000 to 175,000,000 Rials in the sample, were normalized to a [0,1] range to improve model convergence.
- **Feature Extraction**: The ProbabilisticFraudWalk algorithm was applied to extract temporal sequence features, and source/target neighbor sequences were constructed to capture user interaction patterns.

IV. valuation Protocol

The dataset was divided into training and testing sets with varying ratios (30%, 60%, and 90% for training, with the remainder used for testing). Additionally, 5-fold cross-validation was employed to validate the robustness and generalization capability of the proposed model. This rigorous approach helps minimize bias, and provides more reliable results.

TABLE 4
Example of Real-World Transaction Dataset

Primary_ID	TRNS_Type	Date	Time	Amount	Terminal_Ty	Terminal_ID	Second_ID
992e7ca6cc0d738d85b2	10	7/28/2022 0:00	1.42E+08	872000	7	13ae6270758cdba26ec1	8457468be95f8a197756
737b3293b0d5ade9ad90	10	5/6/2022 0:00	1.36E+08	1700000	1	f52c0e50a8eb31e3965c	9d53d2f9c5fe1fc7f260
737b3293b0d5ade9ad90	10	7/8/2022 0:00	1.75E+08	3400000	1	f52c0e50a8eb31e3965c	9d53d2f9c5fe1fc7f260

TABLE 5
Real-World Dataset Description

Feature	Value	Description
Dataset Name	Iranian Dataset	Card-to-card transactions from an Iranian bank in late July 2022
Total Number of Transactions	1,048,575	Total recorded transactions over 15 days
Unique Primary_IDs	338,987	Number of unique source accounts (sender cards)
Unique Second_IDs	16,965	Number of unique destination accounts (receiver cards)
Number of Normal Transactions	1,046,682	Transactions linked to normal accounts (based on Second_ID labels)
Number of Fraudulent Transactions	1,893	Transactions linked to fraudulent accounts (based on Second_ID labels)
Number of Features	8	Attributes: Primary_ID, Second_ID, TRNS_Type, Date, Time, Amount, Terminal_Type, Terminal_ID
Class Imbalance Ratio	0.11% (Fraudulent)	Proportion of fraudulent transactions relative to total transactions
Temporal Coverage	15 days	Data collection period in late July 2022
Terminal Types	POS, Online, ATM	Types of terminals used for transactions

5.2. Benchmark datasets

We utilized two real-world datasets presented by Pozzolo et al. [25] and the IEEE CIS fraud dataset, provided by Vesta, to evaluate the proposed method.

The primary dataset consists of transaction records generated by European cardholders in September 2013 for two days. The complexity of this dataset comes from its significant class imbalance structure. Each individual transaction is represented by a set of 30 features, of which 28 are provided after applying the PCA transformation ($v_1, v_2, \ldots, v_{28}$). The “time” and “amount” features remain unchanged, representing the time gap between the current and initial transactions as well as the value of the current transaction made by the cardholder, respectively. In addition, a “class” label is added, indicating the nature of the transaction as fraudulent with a label “1” or normal with a label “0”.

The second dataset originates from a prominent Brazilian bank and covers the period from April 14th to September 12th, 2004. Each transaction in this dataset is characterized by 17 numerical features and is also labeled with a “class” label, either “S” to indicate a fraudulent transaction or “N” to indicate a normal transaction [26].

Additional information about the two datasets is given in Table 6.

TABLE 6
Descriptions of the benchmark datasets

Dataset	#Normal	#Fraudulent	#Features	#Samples
European dataset	284315	492	30	284807
Brazilian dataset	360792	14031	17	374823

Our experiments are conducted on the data partitioned into a training set and a test set. The results presented in this section are exclusively derived from the test set. Additionally, to ensure the robustness of the proposed model, we employed 5-fold cross-validation across all our experiments, thus validating its consistency.

The European dataset has a very low fraud rate (0.172% of all transactions), and the Brazilian dataset is more balanced (3.74% fraud). These differences highlight the need to evaluate a model across different levels of class imbalanced. In the proposed method, oversampling methods are used to balance the positive (normal) and negative (fraud) samples.

5.3 Evaluation metrics

5.3.1 F1-score

The F1-score is a measure that combines both precision and recall metrics into a composite value. This score is obtained from the harmonic mean of precision and recall and provides a balanced view of the performance of the model. The F1-score is particularly useful for dealing with unbalanced datasets where one class outperforms the other. This criterion is calculated using the following formula:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

The F1-score ranges from 0 to 1, and its higher values indicate better model performance

5.3.2 Macro-F1

The Macro F1 Score is computed through an approach wherein the F1 score is independently calculated for each class and subsequently averaged. This methodology operates under the assumption that all classes bear equal significance, although this assumption may not universally hold true. The formula for the Macro F1 Score is expressed as:

$$Macro\ F1 = \frac{1}{N}\sum_{i=1}^{N} F1_i$$

Here, N represents the number of classes, and $F1_i$ denotes the F1 score for each individual class.

5.3.3 Micro-F1

In the micro approach, the summation of contributions from all classes is employed to calculate the average F1 score. This methodology proves beneficial in instances where there exists a class imbalance within the dataset, as it ensures that smaller classes are accorded equal importance to larger ones. The formula for the micro F1 score can be derived as:

$$\text{Micro F1} = 2 \times \frac{Precision_{micro} \times Recall_{micro}}{Precision_{micro} + Recall_{micro}}$$

5.3.4 AUC

The AUC (Area Under the Curve) is calculated based on the Receiver Operating Characteristic (ROC) curve. The ROC curve is created by plotting the true positive rate (sensitivity) against the false positive rate (1 - specificity) at various threshold settings. The AUC is then computed as the area under this ROC curve. For a more formal explanation, let's denote:

TPR as the true positive rate (sensitivity) and FPR as the false positive rate (1 - specificity).

The AUC is calculated by integrating the ROC curve. In practice, this integration is often approximated using methods like the trapezoidal rule. The formula for AUC can be expressed as:

$$AUC = \int_0^1 TPRdFPR$$

In discrete terms, if the pairs of (FPR_{i}, TPR_i) have been calculated for different threshold settings, we can use the trapezoidal rule:

$$AUC \approx \sum_{i=1}^{n} \frac{(TPR_i + TPR_{i-1}).(FPR_{i-1} - FPR_i)}{2}$$

Here, n is the number of thresholds considered.

5.4 Experiments on benchmark datasets

As mentioned by Jurgovsky et al., the LSTM model is emerging as the leading technique for fraud detection [27]. The performance of this model surpasses conventional machine learning approaches and exhibits significant improvements. Consequently, we chose the LSTM model as the baseline for all our experiments.

The Table 7 and Table 8 presents the performance metrics of five different models, GRU, LSTM, Phased LSTM, and the two proposed model as Cluster-LSTM and Cluster-weighted LSTM, across varying training sizes for European and Brazilian dataset respectively.

In terms of F1-score, the proposed model consistently outperforms the other models across all training sizes, showcasing its effectiveness in capturing both precision and recall. The Macro-F1 and Micro-F1 scores also exhibit similar trends, with the Cluster-weighted LSTM model demonstrating a notable advantage. The area under the curve (AUC) values, which measure the models' ability to distinguish between positive and negative instances, further support the superiority of the Cluster-weighted LSTM model, consistently achieving the highest scores. These results suggest that our two proposed models excel in fraud detection tasks, offering a robust and well-balanced performance across different training sizes.

Upon closer inspection of the results, it is evident that our two models maintain a competitive edge in F1-score metrics, indicating its proficiency in achieving a balance between precision and recall. The Macro-F1 scores highlight the model's ability to generalize well across different classes, while the Micro-F1 scores emphasize its performance at the instance level. Additionally, the AUC values reveal our proposed models superior discriminatory power, showcasing its effectiveness in distinguishing fraudulent and non-fraudulent instances. This consistent and superior performance across diverse training sizes underscores the reliability and adaptability of our models, making them a compelling choice for fraud detection applications

5.5 Experiments on a real-world dataset

The evaluation results of the proposed method compared to other algorithms for three cases of 30%, 60%, and 90% training data and the case of the 5-fold cross-validation algorithm are shown in Table 9.

TABLE 7
Results of Different Models on Various Training Sizes for European dataset

Training Size	Metrics	GRU	LSTM	Phased LSTM	Cluster-LSTM	Cluster-weighted LSTM
0.9	F1-score	0.8302	0.73068	0.71068	0.8359	0.8420
	Macro-F1	0.3346	0.3346	0.3246	0.3338	0.3401
	Micro-F1	0.503	0.50302	0.50102	0.5117	0.5181
	AUC	0.9223	0.883	0.983	0.9302	0.9325
0.7	F1-score	0.8367	0.834	0.8302	0.8384	0.8399
	Macro-F1	0.3345	0.3345	0.2345	0.3475	0.3538
	Micro-F1	0.5026	0.5026	0.2026	0.5103	0.5181
	AUC	0.9107	0.9061	0.8761	0.9194	0.9260
0.5	F1-score	0.8393	0.7242	0.7842	0.888	0.8947
	Macro-F1	0.334	0.334	0.294	0.3421	0.3474
	Micro-F1	0.5016	0.50169	0.49169	0.5119	0.5174
	AUC	0.9164	0.8793	0.9793	0.9794	0.9865
0.3	F1-score	0.7282	0.6355	0.5855	0.7636	0.7711
	Macro-F1	0.3339	0.3339	0.3139	0.3423	0.3497
	Micro-F1	0.5013	0.5013	0.4913	0.5125	0.5205
	AUC	0.7668	0.7044	0.9044	0.9236	0.9293

TABLE 8
Results of Different Models on Various Training Sizes for Brazilian dataset

Training Size	Metrics	GRU	LSTM	Phased LSTM	Cluster-LSTM	Cluster-weighted LSTM
0.9	F1-score	0.8821	0.8361	0.8261	0.8889	0.8896
	Macro-F1	0.3782	0.3831	0.3431	0.3773	0.3861
	Micro-F1	0.5234	0.5282	0.5222	0.5347	0.5421
	AUC	0.9512	0.9338	1.0078	0.9672	0.9615
0.7	F1-score	0.8842	0.8845	0.8398	0.9004	0.9037
	Macro-F1	0.3876	0.3955	0.2842	0.4042	0.4148
	Micro-F1	0.5256	0.5236	0.2556	0.5388	0.5471
	AUC	0.9365	0.9327	0.9121	0.9468	0.9593
0.5	F1-score	0.8982	0.7751	0.8341	0.9482	0.9561
	Macro-F1	0.3824	0.3741	0.3421	0.3934	0.4087
	Micro-F1	0.5283	0.5281	0.5181	0.5402	0.5467
	AUC	0.9568	0.9192	1.0193	1.0194	1.0365
0.3	F1-score	0.7781	0.6851	0.6351	0.8632	0.8711
	Macro-F1	0.3827	0.3813	0.3611	0.3945	0.4021
	Micro-F1	0.5288	0.5253	0.5153	0.5365	0.5445
	AUC	0.8768	0.8244	0.9446	0.9638	0.9793

TABLE 9
Results of Different Models on Various Training Sizes for real-world dataset

Training Size	Metrics	GRU	LSTM	Phased LSTM	Cluster-LSTM	Cluster-weighted LSTM
0.9	F1-score	0.7349	0.7379	0.7179	0.7382	0.7425
	Macro-F1	0.3451	0.3451	0.3351	0.3457	0.3485
	Micro-F1	0.5269	0.5269	0.5249	0.5263	0.5352
	AUC	0.8419	0.8430	0.9430	0.9520	0.9625
0.7	F1-score	0.6912	0.7302	0.7583	0.7592	0.7620
	Macro-F1	0.3475	0.3475	0.2475	0.3475	0.3515
	Micro-F1	0.5326	0.5326	0.2326	0.5326	0.5430
	AUC	0.8460	0.8464	0.8164	0.8561	0.8572
0.5	F1-score	0.6943	0.6172	0.7072	0.7115	0.7266
	Macro-F1	0.3505	0.3505	0.3505	0.3685	0.3706
	Micro-F1	0.5396	0.5396	0.5402	0.5494	0.5508
	AUC	0.8587	0.8489	0.9489	0.9581	0.9632
0.3	F1-score	0.2150	0.1201	0.0701	0.2368	0.2368
	Macro-F1	0.3566	0.3566	0.3666	0.3752	0.3892
	Micro-F1	0.5543	0.5543	0.5443	0.5636	0.5756
	AUC	0.8341	0.8419	0.9256	0.9360	0.9402

According to Table 9, the evaluation results showcase the performance of various models, GRU, LSTM, Phased LSTM, and our two proposed model, across different training sizes (0.9, 0.7, 0.5, and 0.3) on the real-world dataset. F1-score, Macro-F1, Micro-F1, and AUC metrics were employed to assess the models' effectiveness in fraud detection.

At a training size of 0.9, the proposed model outperformed other models with the highest F1-score of 0.7425, indicating a strong balance between precision and recall. The AUC of 0.9625 further underscores its robust discriminative ability. Although the Micro-F1 values were comparable, the proposed model demonstrated superior performance in capturing the nuances of fraud instances.

As the training size reduced to 0.7, the proposed model maintained competitive F1-score and AUC values, showcasing its resilience to variations in the dataset size. Conversely, both GRU and LSTM exhibited a decrease in performance, highlighting the proposed model's efficacy in handling smaller training sets.

At a more constrained training size of 0.5, the proposed model continued to exhibit competitive performance, maintaining a higher F1-score and AUC than its counterparts. This suggests that the proposed model is particularly effective in scenarios with limited training data, making it suitable for real-world applications where data availability may be restricted.

When the training size further decreased to 0.3, all models experienced a decline in performance, with the proposed model demonstrating the highest F1-score of 0.2368. Although the reduced dataset size poses challenges, the proposed model still exhibited notable fraud detection capabilities.

In conclusion, the evaluation results emphasize the effectiveness of the proposed model across varying training sizes, outperforming other models in almost all terms. This highlights its potential as a robust solution for fraud detection in credit card transactions, particularly in scenarios with limited training data.

5.6 Optimal number of clusters

Choosing the best value for k (the number of clusters) is important for K-means because the number of clusters can have a large impact on the performance of the algorithm. To choose the best value of k for the K-means clustering method, the distortion method is used, which is one of the common methods for choosing the best number of clusters (k) in the K-means algorithm. This method is known as a simple and conceptual criterion for choosing k and is based on the amount of data variance within the clusters. As seen in Figure 4, the best value for k is 5.

5.7. Effect of cluster count on metrics

The European dataset is more imbalanced compared to the Brazilian dataset, with the proportion of fraud samples in the European dataset being 0.172% of all transactions,

while in the Brazilian dataset, fraud constitutes 3.74% of all transactions. Furthermore, we demonstrate the impact of the number of clusters on evaluation metrics for each of the datasets.

In the European dataset, as cluster count increases, the number of fraud samples exists in each cluster compared to normal ones become significantly lower and the precision decreases for higher cluster count than 3. This reduction occurs because the number of fraud samples decreases drastically in the clusters, causing the model to bias towards normal ones and fail to correctly identify fraud samples, resulting in a decrease in precision. Consequently, the recall increases as shown in Figure 5.

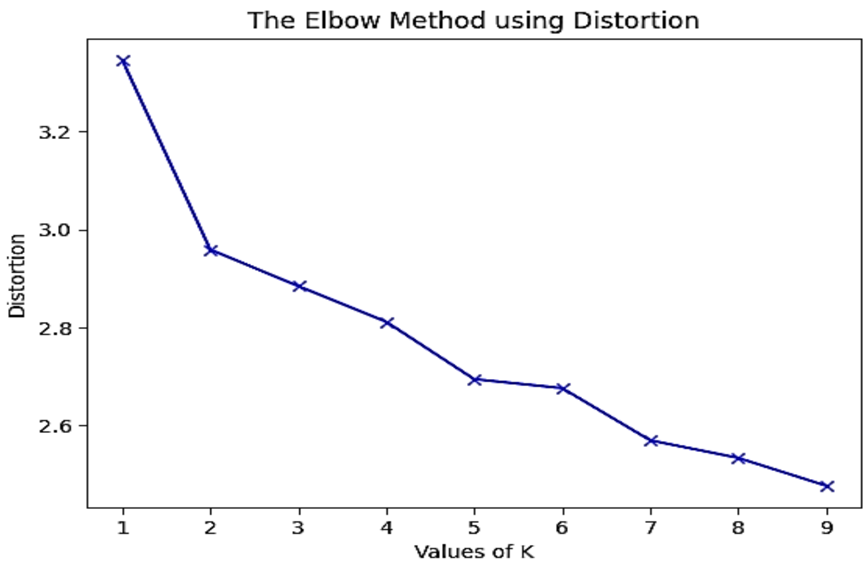


Figure 4. Different k values versus distortion values.

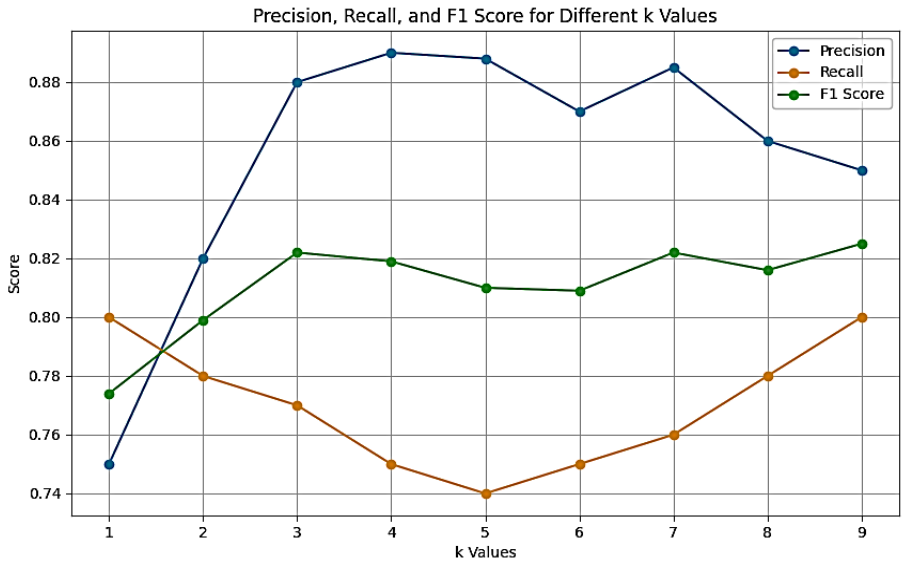


Figure 5. Values of precision, recall, and F1-score for different values of k for European dataset.

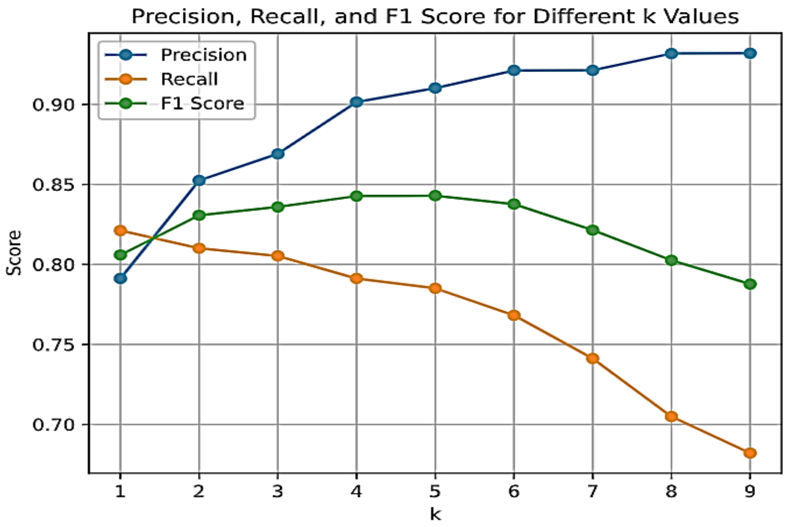


Figure 6. Values of precision, recall, and F1-score for different values of k for Brazilian dataset.

Figure 6 shows the precision, recall, and F1-score values of the proposed method for different values of k on Brazilian dataset. In the Brazilian dataset, due to its higher balance compared to the European dataset, the precision values increase up to the number of clusters k=5 and then their growth rate decreases. As can be seen in Figure 6, as the number of clusters increases, the precision also increases. This increase in precision occurs at a faster rate up to k=5, and from this point onward, the growth rate decreases because the sensitivity of the LSTMs to the samples is increased and overtraining occurs, thus reducing the recall due to the generalization decrease. On the other hand, with the reduction of the number of clusters, the information conflict of the samples increases, the precision decreases, and the recall also increases due to the increase in generalization.

6. Conclusions

In this study, we introduce a novel approach for credit card fraud detection, leveraging a fusion of clustering techniques and a weighted Long Short-Term Memory (LSTM) model. Our experimental results underscore the significant superiority of this proposed method over comparative approaches GRU, LSTM, PhasedLSTM, specifically showcasing enhanced F1-score, Macro-F1, Micro-F1 and AUC metrics. The efficacy of our method stems from its ability to extract clustered information from the dataset, mitigating information conflicts during the learning process.Rigorous experimentation on a real-world dataset of card-to-card transactions highlights the superior performance of our proposed method, particularly excelling when trained on substantial datasets ranging from 60% to 90% of the total data. The method's notable strengths are particularly evident in F1-score, Macro-F1, Micro-F1 and AUC criteria.

The observed trends across varying training sizes underscore the adaptability and robustness of the proposed model, positioning it as a versatile solution for fraud detection in dynamic scenarios. The adaptability of the proposed method in accommodating varying cluster sizes further enhances its practicality. With its potential to significantly bolster fraud detection in credit card transactions, this method holds promise for organizations and banks, promising heightened security and customer trust. For future work, the utilization of complex ensemble approaches, such as boosting or bagging, can be explored for combining multiple LSTM models to improve performance and accuracy.

References

[1] H. Fanai et al. (2023). A novel combined approach based on deep Autoencoder and deep classifiers for credit card fraud detection. *Expert Systems with Applications*. [Online]. *217*, p. 119562. Available: https://doi.org/10.1016/j.eswa.2023.119562

[2] A. Abdallah et al. (2016). Fraud detection system: A survey. *Journal of Network and Computer Applications*. [Online]. *68*, pp. 90–113. Available: https://doi.org/10.1016/j.jnca.2016.04.007

[3] S. Motie et al. (2023). Financial fraud detection using graph neural networks: A systematic review. *Expert Systems with Applications*. [Online]. *240*, p. 122156. Available: https://doi.org/10.1016/j.eswa.2023.122156

[4] E. Esenogho et al. (2022). A neural network ensemble with feature engineering for improved credit card fraud detection. *IEEE Access*. [Online]. *10*, pp. 16400–16407. Available: https://doi.org/10.1109/ACCESS.2022.3148298

[5] W. Hilal et al. (2022). Financial fraud: A review of anomaly detection techniques and recent advances. *Expert Systems with Applications*. [Online]. *193*, p. 116429. Available: https://doi.org/10.1016/j.eswa.2021.116429

[6] Y. Tian et al. (2023). Transaction fraud detection via spatial-temporal-aware graph transformer. *arXiv preprint arXiv:2307.05121*. [Online]. Available: https://arxiv.org/abs/2307.05121
[7] D. V. Kute et al. (2021). Deep learning and explainable artificial intelligence techniques applied for detecting money laundering – a critical review. *IEEE Access*. [Online]. *9*, pp. 82300–82317. Available: https://doi.org/10.1109/ACCESS.2021.3086230
[8] A. N. Eddin et al. (2021). Anti-money laundering alert optimization using machine learning with graphs. *arXiv preprint arXiv:2112.07508*. [Online]. Available: https://arxiv.org/abs/2112.07508
[9] Y. Bao et al. (2022). Artificial intelligence and fraud detection. Innovative Technology at the Interface of Finance and Operations: Volume I. [Online]. pp. 223–247. Available: https://doi.org/10.1007/978-3-030-75729-8_8
[10] B. Wu et al. (2023). Advancement of management information system for discovering fraud in master card based intelligent supervised machine learning and deep learning during SARS‑CoV2. *Information Processing & Management*. [Online]. *60*, 103231. Available: https://doi.org/10.1016/j.ipm.2022.103231
[11] Y. Xie et al. (2022). Learning transactional behavioral representations for credit card fraud detection. IEEE Transactions on Neural Networks and Learning Systems. [Online]. *35(4)*, pp. 5735–5748. Available: https://doi.org/10.1109/TNNLS.2022.3208967
[12] J. F. Roseline et al. (2022). Autonomous credit card fraud detection using machine learning approach. *Computers and Electrical Engineering.* [Online]. *102*, p. 108132. Available: https://doi.org/10.1016/j.compeleceng.2022.108132
[13] J. Guo et al, “Learning sequential behavior representations for fraud detection,” in *Proceeding IEEE International Conference Data Mining (ICDM)*, 2018, pp. 127–136. Available: https://doi.org/10.1109/ICDM.2018.00028
[14] A. Agarwal et al. (2021). Hybrid CNN‑BiLSTM‑attention based identification and prevention system for banking transactions. *NVEO – Natural Volatiles & Essential Oils Journal*. [Online]. *7*, pp. 2552–2560.
[15] B. Fakiha. (2023). Forensic credit card fraud detection using deep neural network. *Journal of Southwest Jiaotong University*. [Online]. *58(1)*, pp. 1–20. Available: https://doi.org/10.35741/issn.0258-2724.58.1.33
[16] G. Zioviris. (2022). Credit card fraud detection using a deep learning multistage model. *The Journal of Supercomputing*. [Online]. *78*, pp. 14571–14596. Available: https://doi.org/10.1007/s11227-022-04465-9
[17] Raval, J., Bhattacharya, P., Jadav, N. K., Tanwar, S., Sharma, G., Bokoro, P. N., Elmorsy, M., Tolba, A., & Raboaca, M. S. (2023). RaKShA: A trusted explainable LSTM model to classify fraud patterns on credit card transactions. *Mathematics*. [Online]. *11*, p. 1901. Available: https://doi.org/10.3390/math11081901
[18] D. Wang et al., “A semi‑supervised graph attentive network for financial fraud detection,” in *Proceeding IEEE International Conference Data Mining (ICDM)*, 2019, pp. 598–607. https://doi.org/10.1109/ICDM.2019.00070
[19] Y. Jiang et al. (2022). Telecom fraud detection via Hawkes‑enhanced sequence model. *IEEE Transactions on Knowledge and Data Engineering*. [Online]. *35*, pp. 5311–5314. Available: https://doi.org/10.1109/TKDE.2022.3150803
[20] Y. Liu et al. (2023). Improving fraud detection via hierarchical attention‑based graph neural network. *Journal of Information Security and Applications*. [Online]. *72*, p. 103399. Available: https://doi.org/10.1016/j.jisa.2022.103399
[21] L. Wang et al., "Modeling heterogeneous graph network on fraud detection: a community-based framework with attention mechanism," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 1959-1968. Available: https://doi.org/10.1145/3459637.3482277
[22] G. Liu et al. (2020). Fraud detection via behavioral sequence embedding. *Knowledge and Information Systems*. [Online]. *62*, pp. 2685–2708. Available: https://doi.org/10.1007/s10115-019-01433-3
[23] O. Imran et al. (2025). Leveraging LSTM and attention for high‑accuracy credit card fraud detection. *Fusion: Practice and Applications*. [Online]. *17(1)*, 209–220.
[24] T. Trinh et al. (2024). Dynamic graph neural networks for multi‑level financial fraud detection: A temporal‑structural approach. *Annals of Applied Sciences*. [Online]. *5(1)*.
[25] A. Dal Pozzolo et al., “Calibrating probability with undersampling for unbalanced classification,” in *Proceeding IEEE symposium series on computational intelligence*, 2015, pp. 159–166.
[26] M. F. A. Gadi et al., “Credit card fraud detection with artificial immune system,” in *international conference on artificial immune systems*, Berlin, Heidelberg, 2008, pp. 119–131. Availae: https://doi.org/10.1007/978-3-540-85072-4_11
[27] J. Jurgovsky et al. (2018). Sequence classification for credit‑card fraud detection. *Expert Systems with Applications*. [Online]. *100*, pp. 234–245. Available: https://doi.org/10.1016/j.eswa.2018.01.037